



Securing the Web with the Next-Generation Public-Key Cryptosystem

Hans Eberle, Vipul Gupta,

Sheueling Chang, Nils Gura

Sun Microsystems Laboratories

SNRC Industry
Seminar

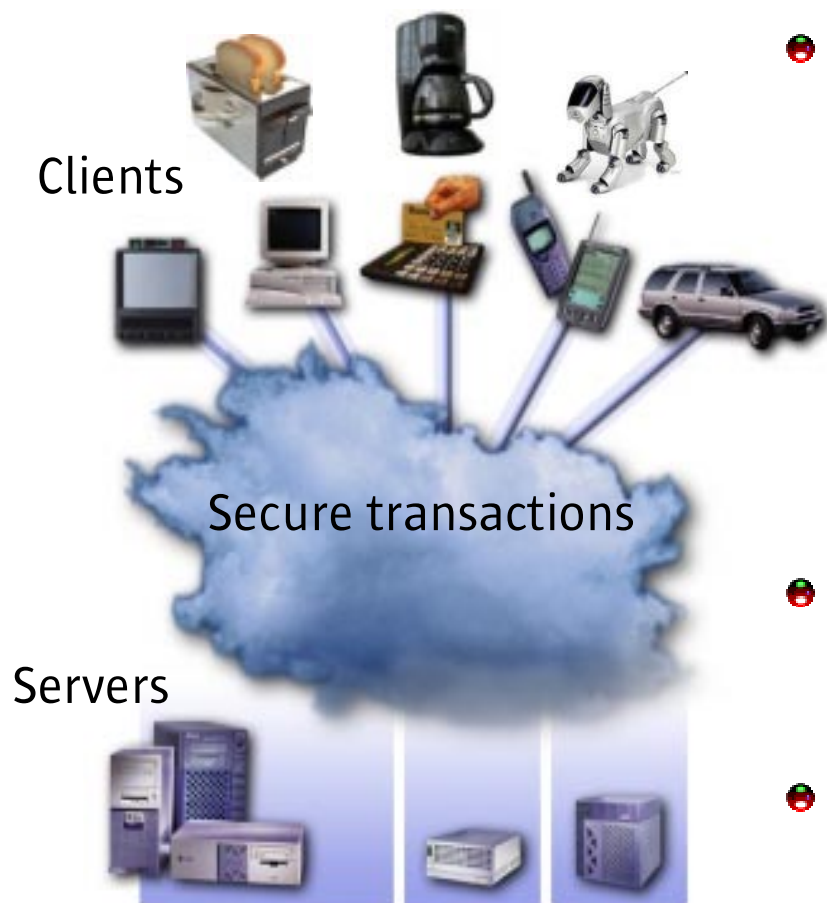
Oct 14, 2003



Talk Outline

- Project background
- Elliptic Curve Cryptography (ECC) overview
- Seeding industry adoption of ECC
 - IETF standardization in SSL
 - Contributions to OpenSSL/Apache, NSS/Mozilla, Java
- Accelerating public key cryptosystems
 - ECC and RSA accelerators

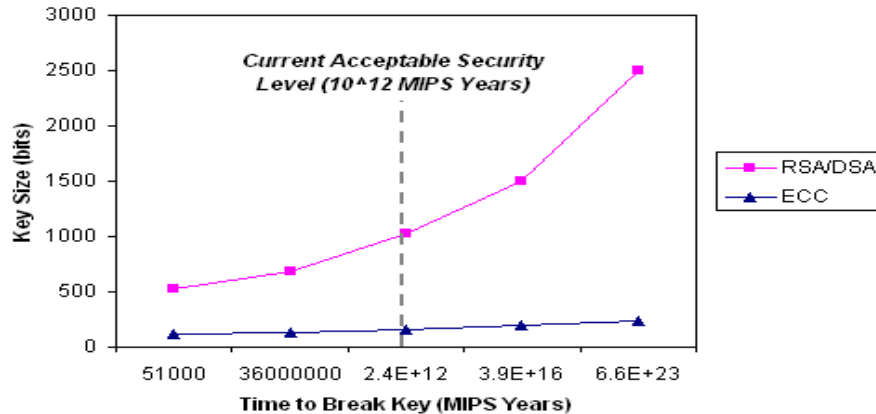
The Next Wave of the Internet



- Trends:
 - Number of devices ↑
 - Device capabilities ↓
 - Security needs ↑
 - Secure transactions ↑
- Clients need **lower capability threshold** for strong crypto
- Servers need **greater capacity** to handle secure transactions

Elliptic Curve Cryptography

COMPARISON OF SECURITY LEVELS of ECC and RSA & DSA



- Computationally efficient public-key cryptosystem, highest security strength per bit

- Memory, power and bandwidth savings (well suited for wireless / constrained devices)

- Advantage improves as security needs increase

- Endorsed / standardized by NIST, ANSI, IEEE, IETF

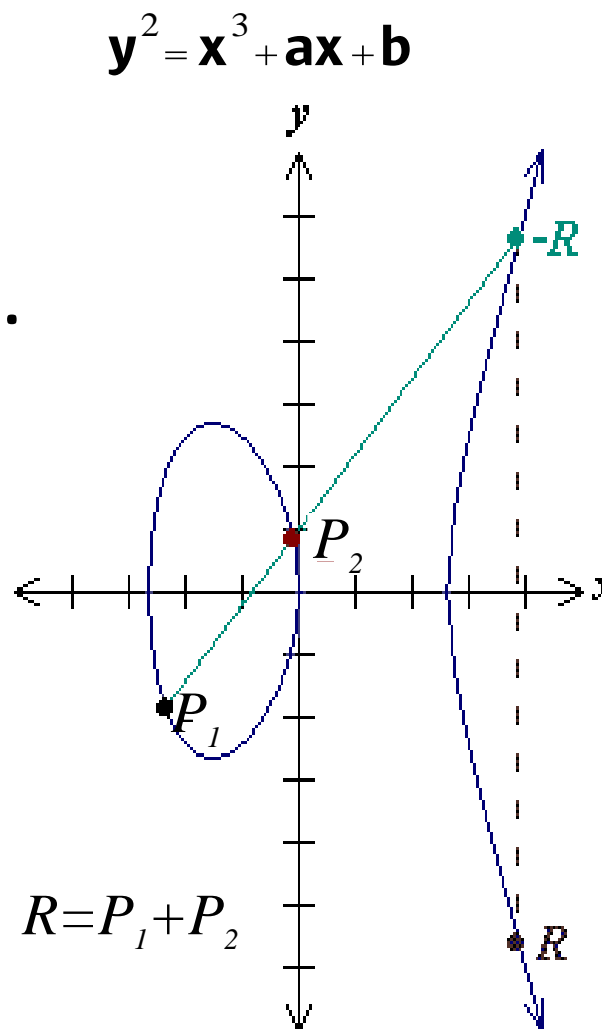
- Good match for AES

Sym.	RSA	ECC	Size	Perf.
80	1,024	160	6:1	4:1
112	2,048	224	9:1	14:1
128	3,072	256	12:1	
192	7,680	384	20:1	
256	15,360	521	30:1	

Equivalent key sizes

Elliptic Curve Cryptography

- Parameters: elliptic curve and a base point P
- Scalar point multiplication: $Q = kP$.
e.g. $11P = ((P*2)*2+P)*2+P$
- Elliptic Curve Discrete Logarithm Problem (ECDLP): given kP (public-key) & P , find k (private-key)
- Best known attack: Pollard-rho algorithm $O(n^{1/2})$



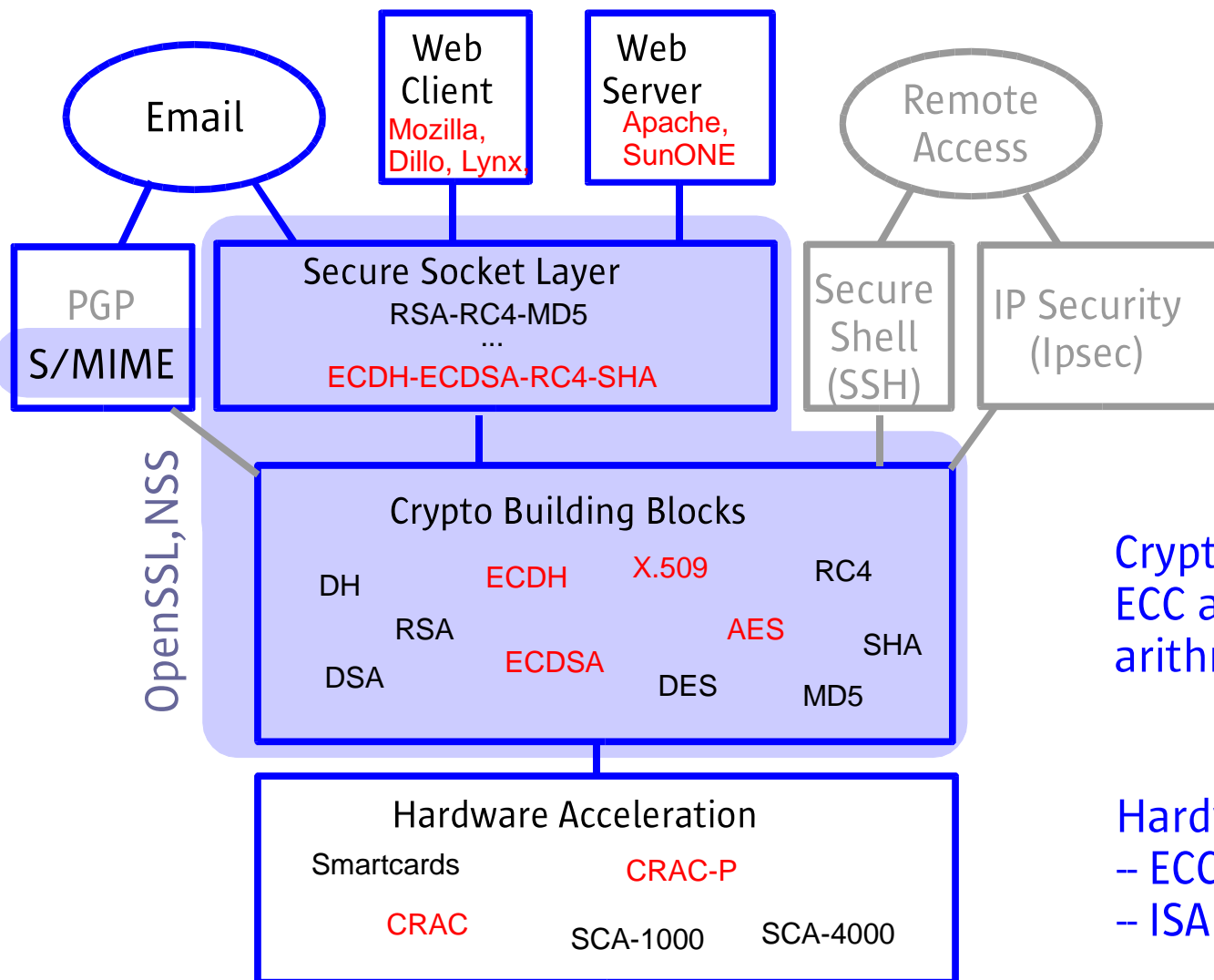
ECC Adoption

- Recommended curves, lower level algorithms (ECDSA, ECDH) already standardized: ANSI X9.62, ANSI X9.63, FIPS 186-2 (Feb 2000)
- ECC supported in standard cryptographic APIs – PKCS#11, Java Card™ 2.2, J2SE 1.5
- ECC usage proposed in IETF protocols: IPsec, S/MIME, TLS
- Some products and open source code available.
- Very much in the early phase (“catch 22” situation)

Project Approach

- Seed market adoption of ECC
 - Standardization efforts
 - Contributions to open source community w/ very liberal licensing
 - Articulate benefits in “real-world” scenarios
- Develop acceleration techniques for ECC and RSA
 - Hardware acceleration through crypto core or ISA extensions
 - Optimized firmware for ECC point multiplication and RSA modular multiplication

Systems Approach



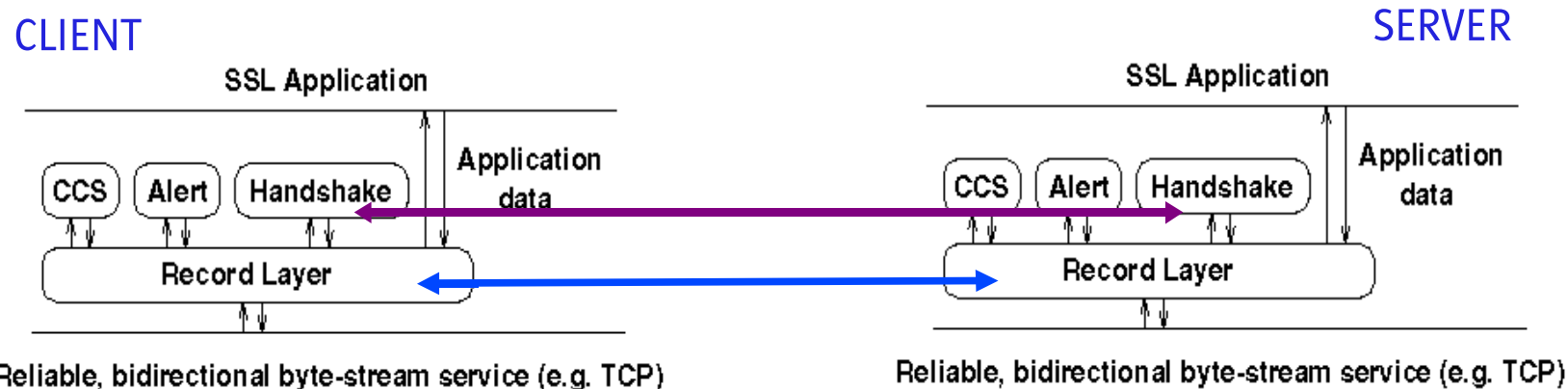
Secure Applications:
ECC-based HTTPS,
Email

Security protocols:
ECC-enabled SSL,
S/MIME

Crypto library:
ECC algorithms, field
arithmetic, and certs

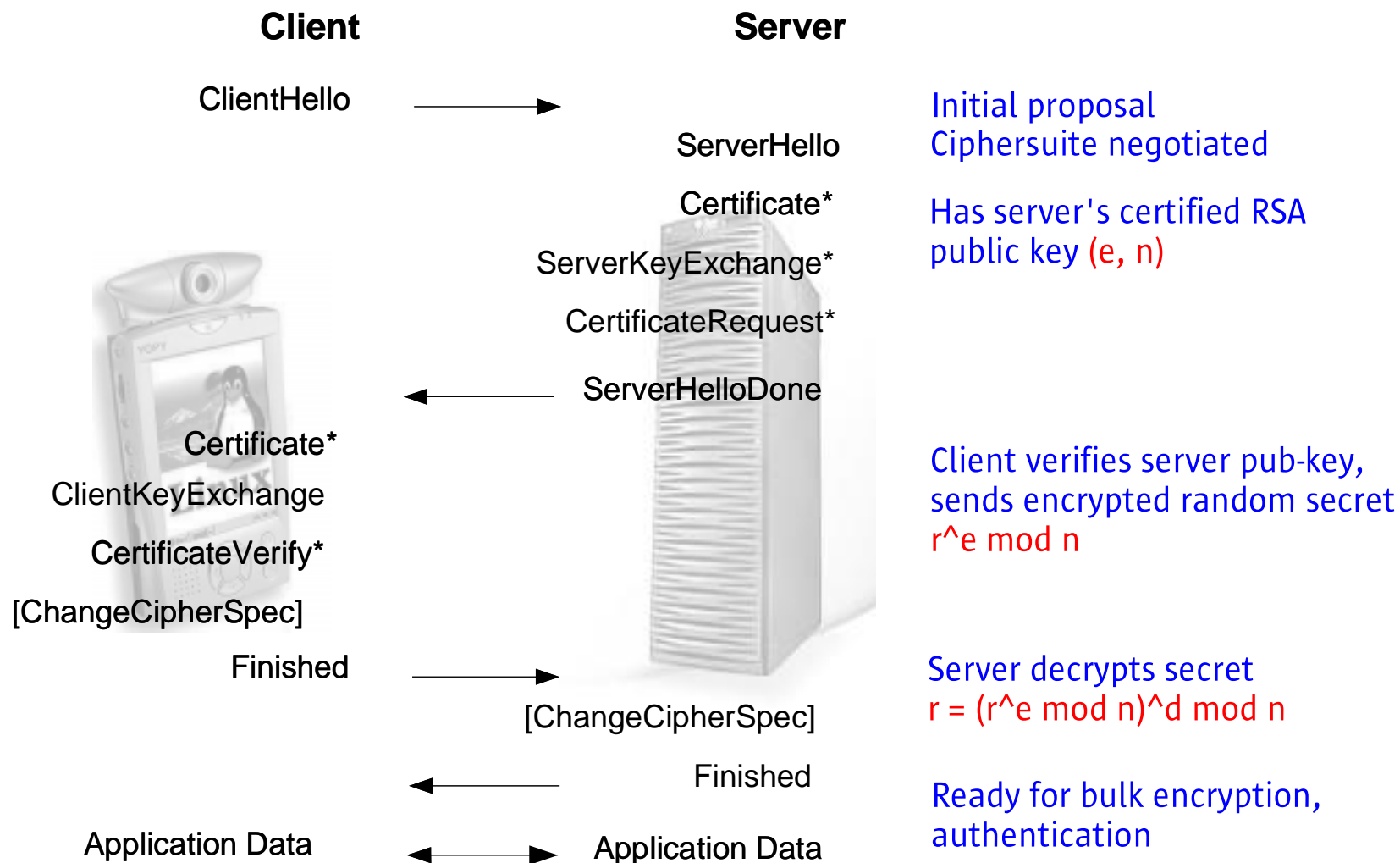
Hardware Acceleration:
– ECC crypto boards,
– ISA ext. for ECC/RSA

Secure Socket Layer Overview



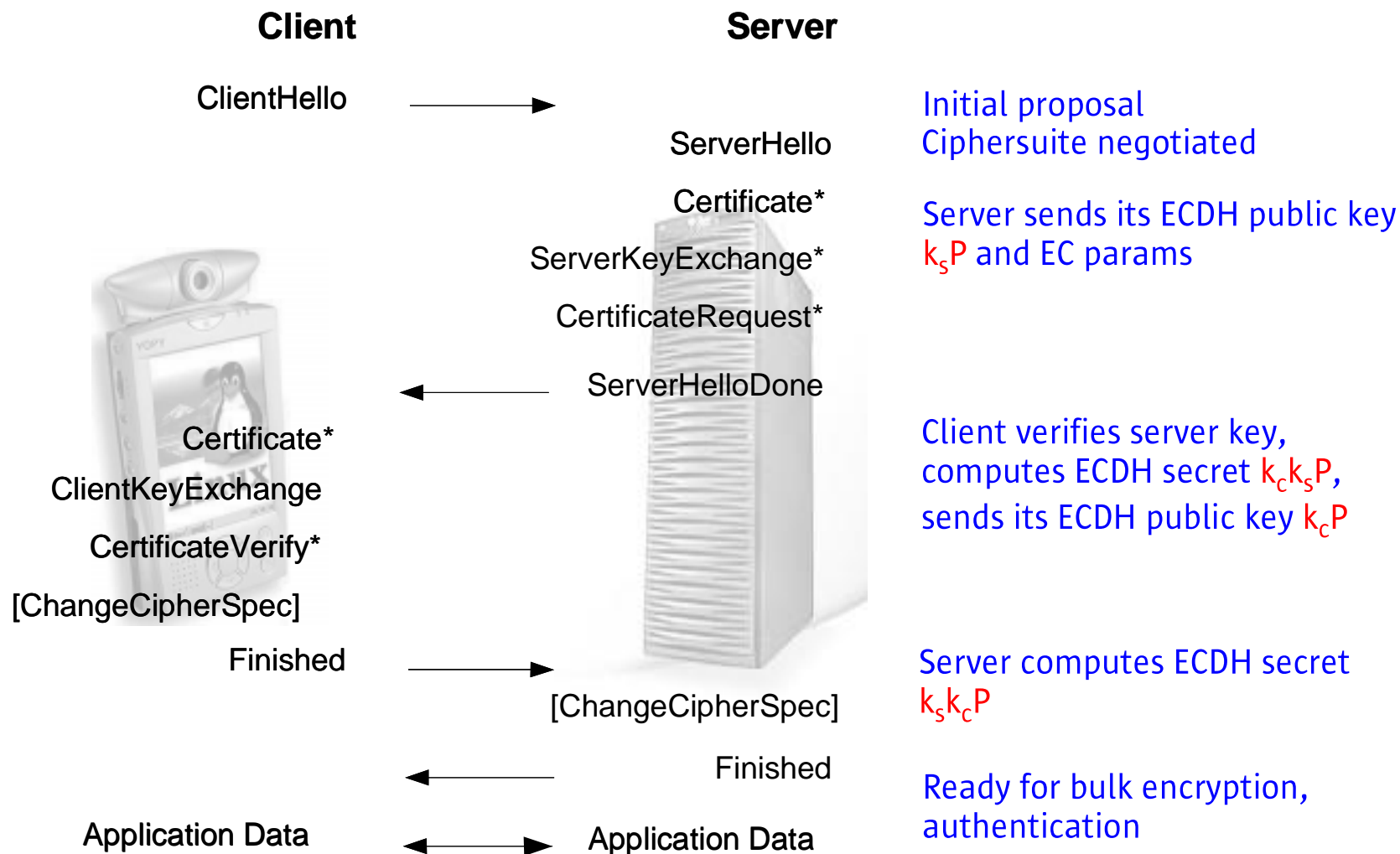
- Handshake protocol uses (expensive) public-key algorithms to establish a shared secret
- Record Layer uses shared secret to perform (cheaper) symmetric-key encryption and MAC on bulk data
- Ciphersuite example: RSA_WITH_RC4_128_MD5

RSA Handshake



Shared secret, r , is essentially transported (determined by client only) 10

ECDH Handshake



Four variants: ECDH-ECDSA, ECDH-RSA, ECDHE-ECDSA, ECDHE-RSA

Performance Impact (ECC vs. RSA)

OpenSSL microbenchmark:

ECC public key operations
are **faster**

current key sizes: **2.4x**

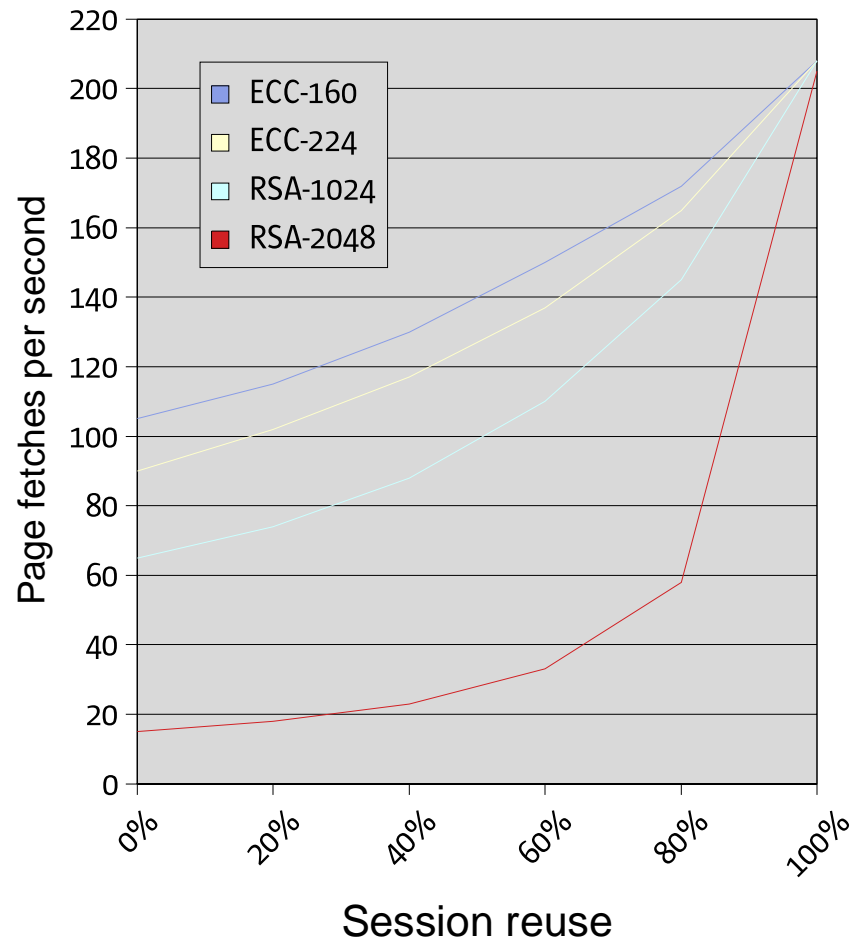
future key sizes: **11x**

Apache SSL benchmark*:

ECC server achieves **higher
secure transaction rate**

current key sizes: **1.3x**

future key sizes: **3.8x**



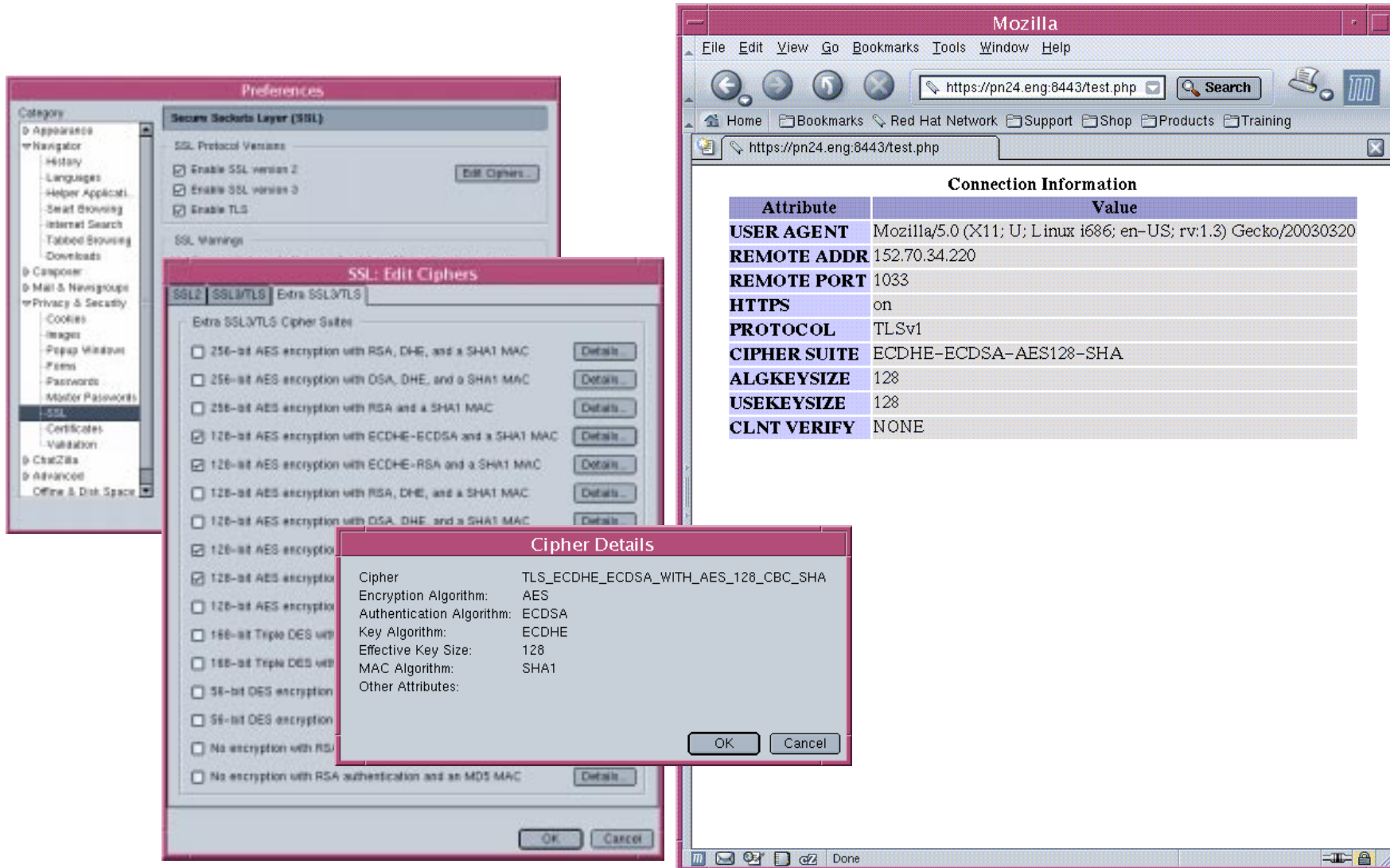
*Apache 2 w/OpenSSL, 1 cpu, 30KB file size, 900MHz
UltraSPARC III, 66% session reuse

Project Highlights (Software)

- Software Development
 - ECC contributions to [OpenSSL 0.9.8](#) and [Netscape Security Services](#) (NSS 3.8 and 3.9*), liberal licensing
 - Integrated ECC with [Apache](#) and [JES web servers](#); [Mozilla](#), [Lynx](#) and [Dillo](#) browsers
- Standardization
 - IETF drafts “ECC cipher suites in TLS” and “TLS Extensions for Elliptic Curve Cipher Suites”*
 - Resolve ECDH inconsistencies in IEEE and PKCS#11*
- Performance analysis (ECC v/s RSA)
- Address security needs for small devices*

* in progress

Mozilla browser communicating securely with an Apache webserver using ECC cipher suites

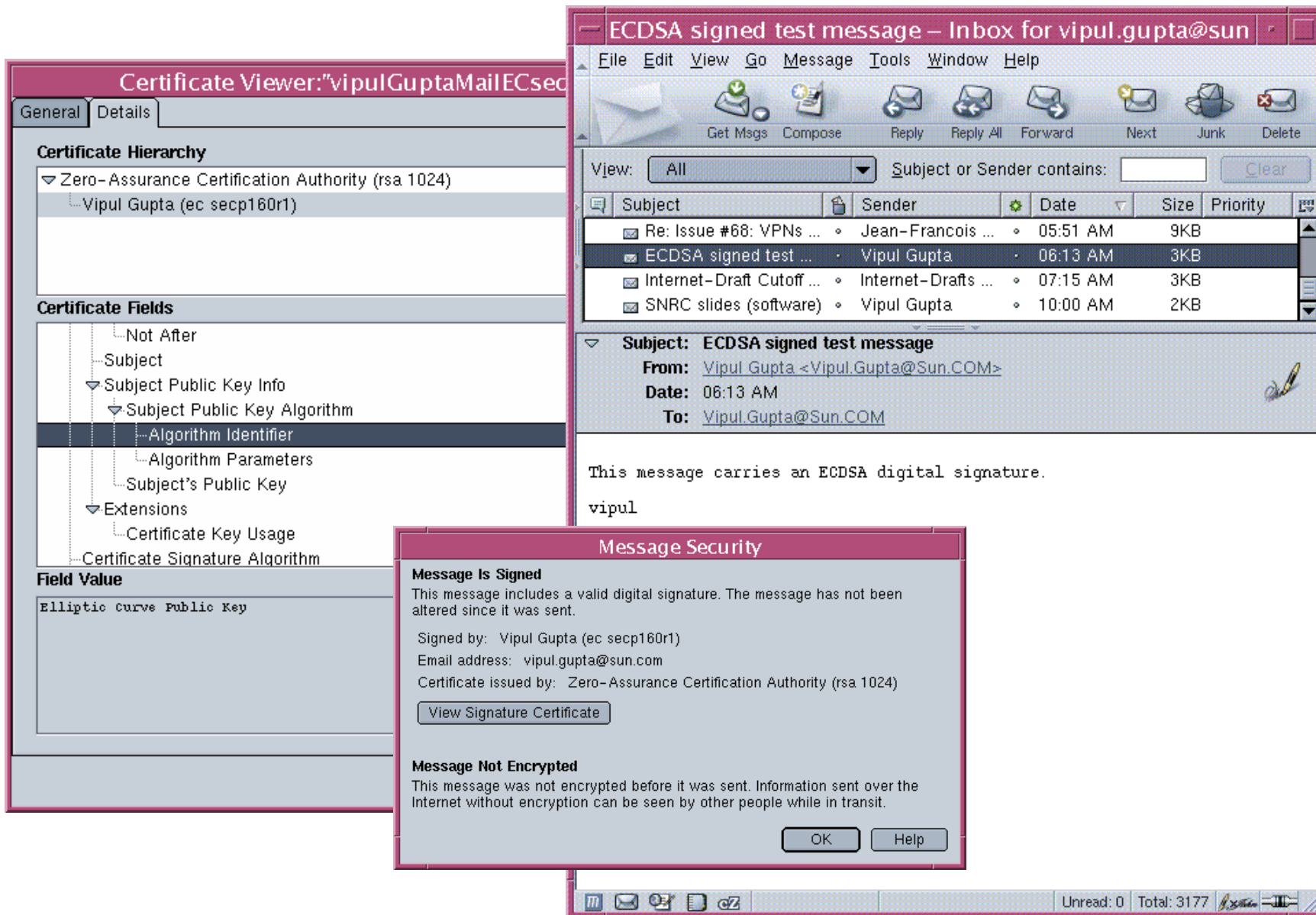


The image shows a composite of three screenshots from the Mozilla browser interface:

- Left Screenshot: Preferences - Secure Sockets Layer (SSL)**
 - SSL Protocol Versions:
 - Enable SSL version 2
 - Enable SSL version 3
 - Enable TLS
 - SSL Warnings: (empty)
- Middle Screenshot: SSL: Edit Ciphers**
 - SSL: SSL/TLS Extra SSL/TLS
 - Extra SSL/TLS Cipher Suite:
 - 256-bit AES encryption with RSA, DHE, and a SHA1 MAC
 - 256-bit AES encryption with DSA, DHE, and a SHA1 MAC
 - 256-bit AES encryption with RSA and a SHA1 MAC
 - 128-bit AES encryption with ECDSA and a SHA1 MAC
 - 128-bit AES encryption with ECDSA-RSA and a SHA1 MAC
 - 128-bit AES encryption with RSA, DHE, and a SHA1 MAC
 - 128-bit AES encryption with DSA, DHE, and a SHA1 MAC
 - 128-bit AES encryption
 - 128-bit AES encryption
 - 128-bit AES encryption
 - 168-bit Triple DES with
 - 168-bit Triple DES with
 - 56-bit DES encryption
 - 56-bit DES encryption
 - No encryption with RSA
 - No encryption with RSA authentication and an MD5 MAC
- Right Screenshot: Mozilla Browser - Connection Information**

Attribute	Value
USER AGENT	Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.3) Gecko/20030320
REMOTE ADDR	152.70.34.220
REMOTE PORT	1033
HTTPS	on
PROTOCOL	TLSv1
CIPHER SUITE	ECDHE-ECDSA-AES128-SHA
ALGKEYSIZE	128
USEKEYSIZE	128
CLNT VERIFY	NONE

Mozilla reading an ECDSA-signed message



The screenshot shows the Mozilla Mail interface with an ECDSA signed message open. The message content is:

This message carries an ECDSA digital signature.
vipul

The message details are:

- Subject:** ECDSA signed test message
- From:** Vipul Gupta <Vipul.Gupta@Sun.COM>
- Date:** 06:13 AM
- To:** Vipul.Gupta@Sun.COM

The Certificate Viewer window shows the following details:

Certificate Hierarchy

- Zero-Assurance Certification Authority (rsa 1024)
 - Vipul Gupta (ec secp160r1)

Certificate Fields

- Not After
- Subject
- Subject Public Key Info
 - Subject Public Key Algorithm
 - Algorithm Identifier
 - Algorithm Parameters
 - Subject's Public Key
- Extensions
 - Certificate Key Usage
 - Certificate Signature Algorithm

Field Value

Elliptic Curve Public Key

The Message Security dialog box indicates:

Message Is Signed
This message includes a valid digital signature. The message has not been altered since it was sent.
Signed by: Vipul Gupta (ec secp160r1)
Email address: vipul.gupta@sun.com
Certificate issued by: Zero-Assurance Certification Authority (rsa 1024)
[View Signature Certificate](#)

Message Not Encrypted
This message was not encrypted before it was sent. Information sent over the Internet without encryption can be seen by other people while in transit.

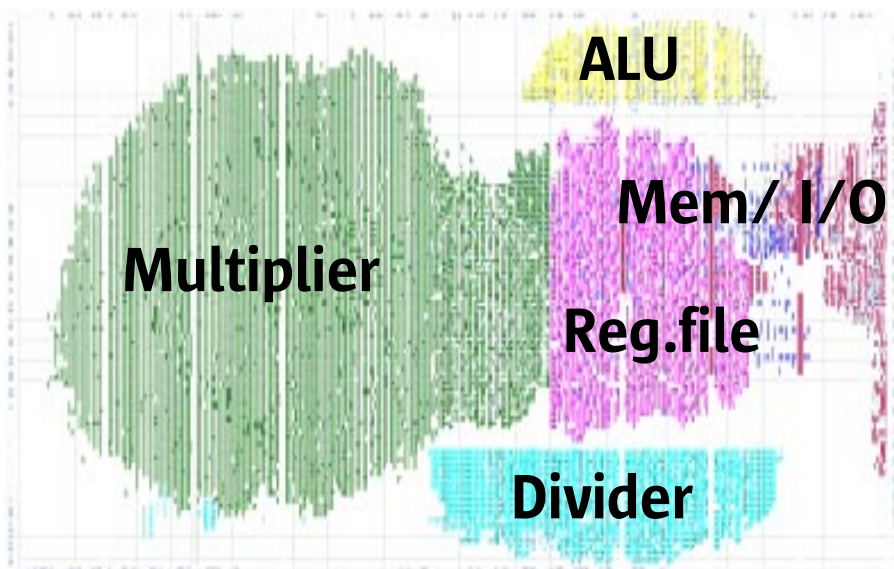
Buttons: OK, Help

Public Key Crypto Math

- RSA is based on **modular exponentiation**
 $C = M^e \bmod n$
- ECC is based on **point multiplication**
 $Q = k \cdot P$
- Modular multiplication on large operands is the key underlying function
- ECC math is more complicated; also requires multiprecision addition/subtraction/division

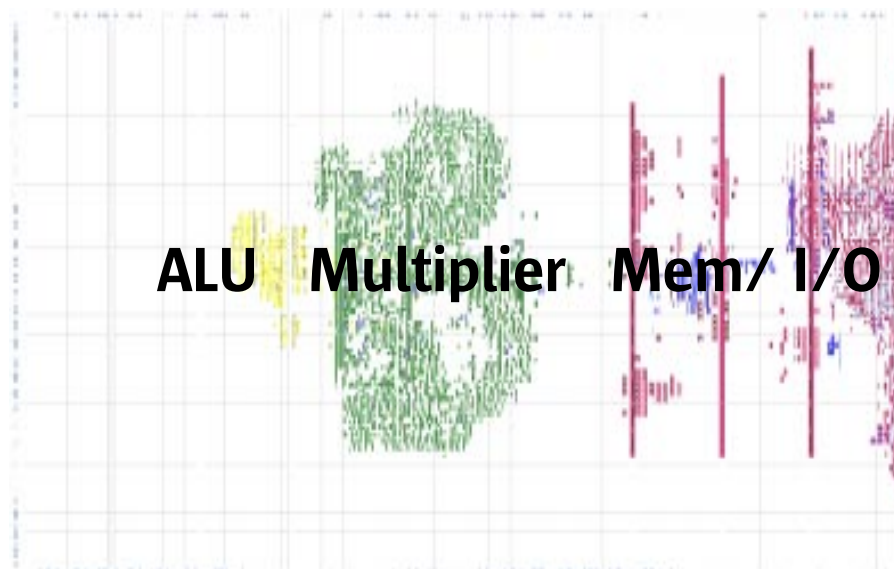
Crypto Accelerator Prototypes

1st Generation ECC Accelerator



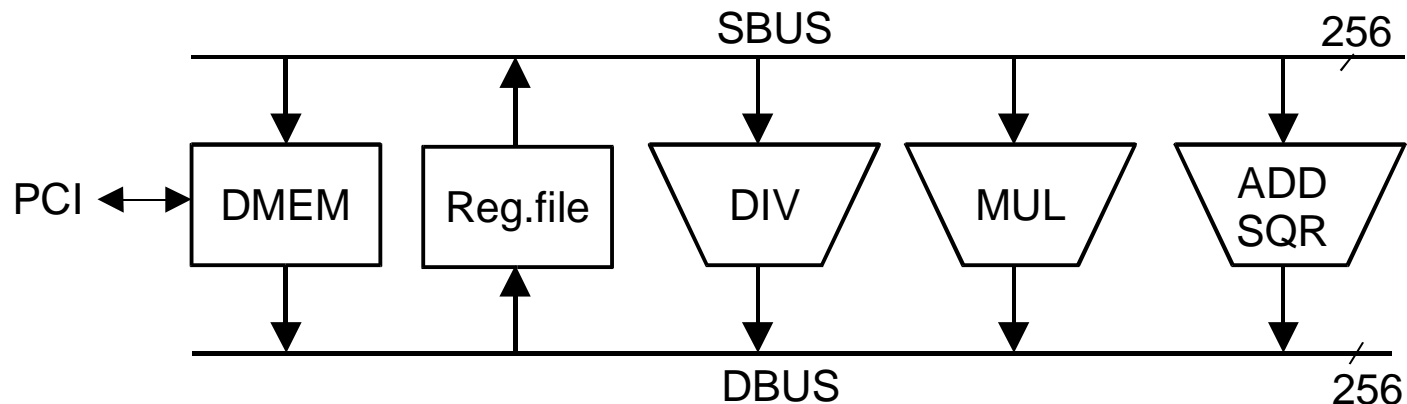
- Dedicated 256-bit coprocessor
- Fastest reported ECC implementation (66 MHz, 3..4-cycle non-pipelined 256x256 mul)
- 6,987 ECC-163 op/s

2nd Generation ECC & RSA Accelerator



- General-purpose 64-bit processor
- Projected performance (1.5 GHz, 2-cycle 64x64 pipelined mul)
 - 10,000 ECC-163 op/s
 - 2,500 RSA-1024 op/s

1st Generation ECC Accelerator



- **Dedicated crypto coprocessor for ECC**

- Binary polynomial fields, key sizes ≤ 255 bits
- Optimized performance for named curves
- Support for generic curves

- **Architecture**

- Microprogrammable 256-bit data path
- Load/store architecture
- ADD, SQR, MUL, DIV

1st Generation Highlights

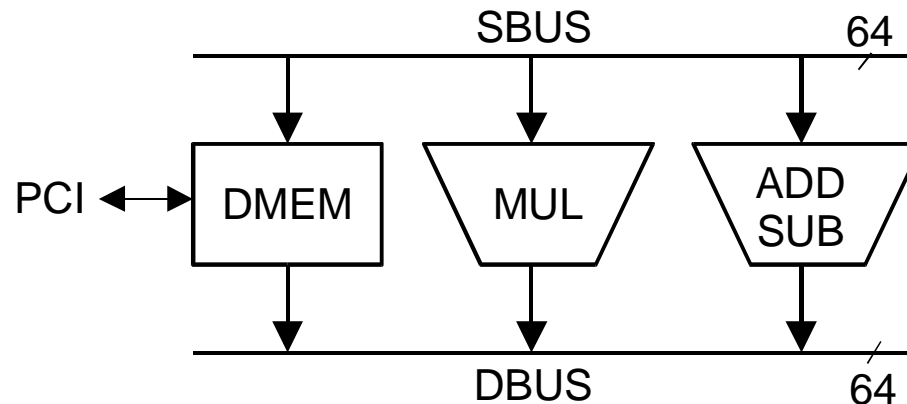
- **Fastest** reported ECC implementation
 - Fast modular multiplier (4 cycles for 256-bit mul)
 - 1-cycle squarer
 - Hardware divider
 - Parallel and overlapped instruction execution
- First ECC implementation supporting multiple **named curves** and **arbitrary generic curves**

Performance

	Hardware	Software*	Speedup
	66 MHz [op/s]	900 MHz USIII [op/s]	
Named Curves			
GF(2 ¹⁶³)	6987	322	21.7x
GF(2 ¹⁹³)	5333	294	18.1x
GF(2 ²³³)	4438	223	19.9x
Generic Curves			
GF(2 ¹⁶³)	3308		
GF(2 ¹⁹³)	2375		
GF(2 ²³³)	1980		

* 64-bit OpenSSL 0.9.8

2nd Generation ECC and RSA Accel.



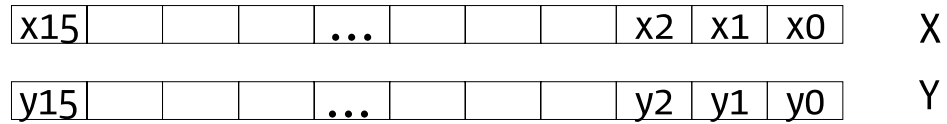
- General-purpose proc. supporting ECC, RSA, DH, DSA
 - Binary polynomial fields and prime fields
 - Arbitrary key sizes, arbitrary curves
- Architecture
 - Microprogrammable 64-bit data path
 - Memory operands
 - Multiprecision MUL, ADD, SUB
 - Dual-issue machine

2nd Generation Highlights

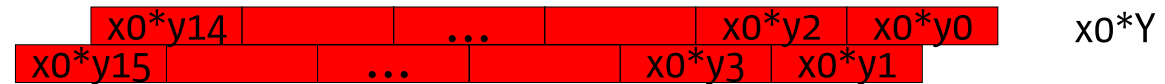
- Instruction set optimized for fast modular multiplication
- Multiprecision arithmetic for arbitrary operand sizes
- Dual-field multiplier supports binary polynomial fields and prime integer fields

Montgomery Multiplication

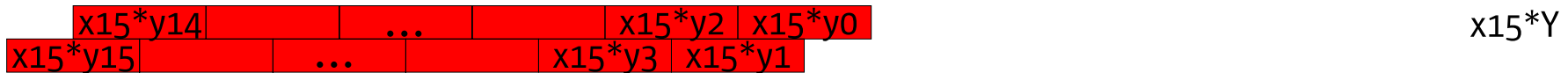
$$u = X * Y * 2^{-k} \pmod{P}$$



1. Step:
Multiprecision
Multiplication

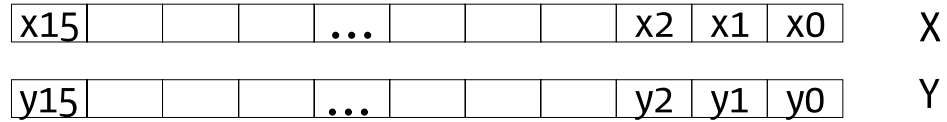


...

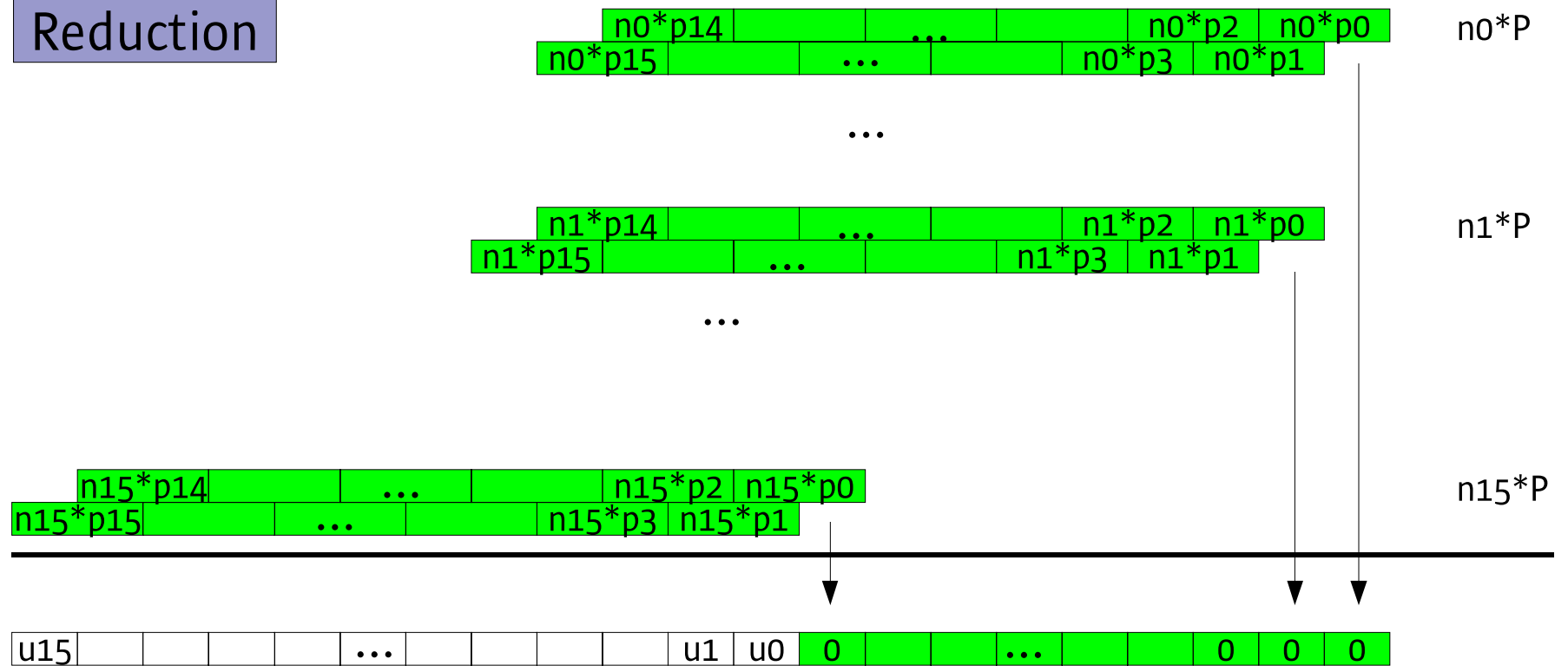


Montgomery Multiplication

$$u = X * Y * 2^{-k} \text{ mod } P$$

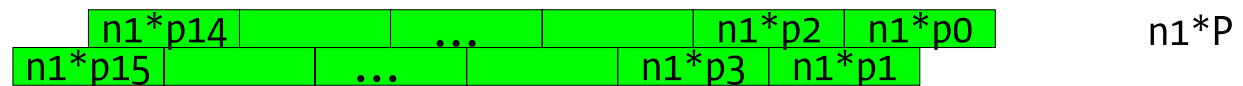
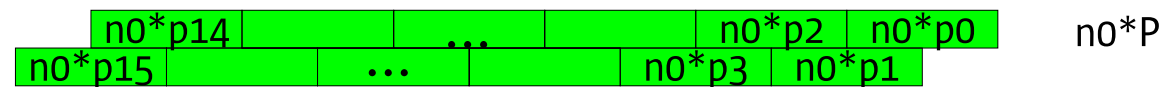
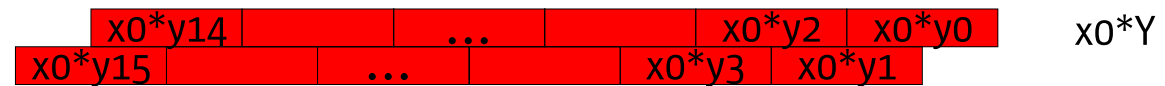
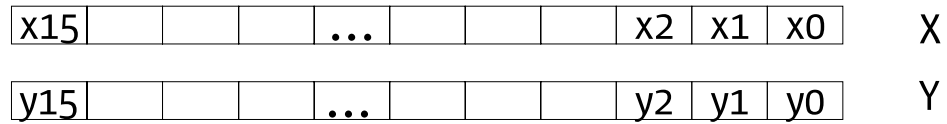


2. Step: Reduction

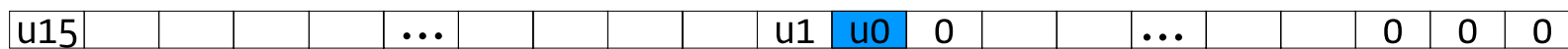
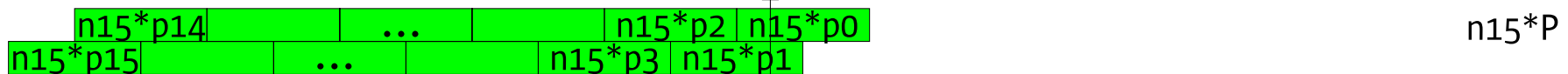
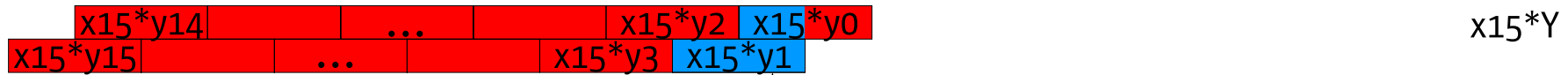


Montgomery Multiplication

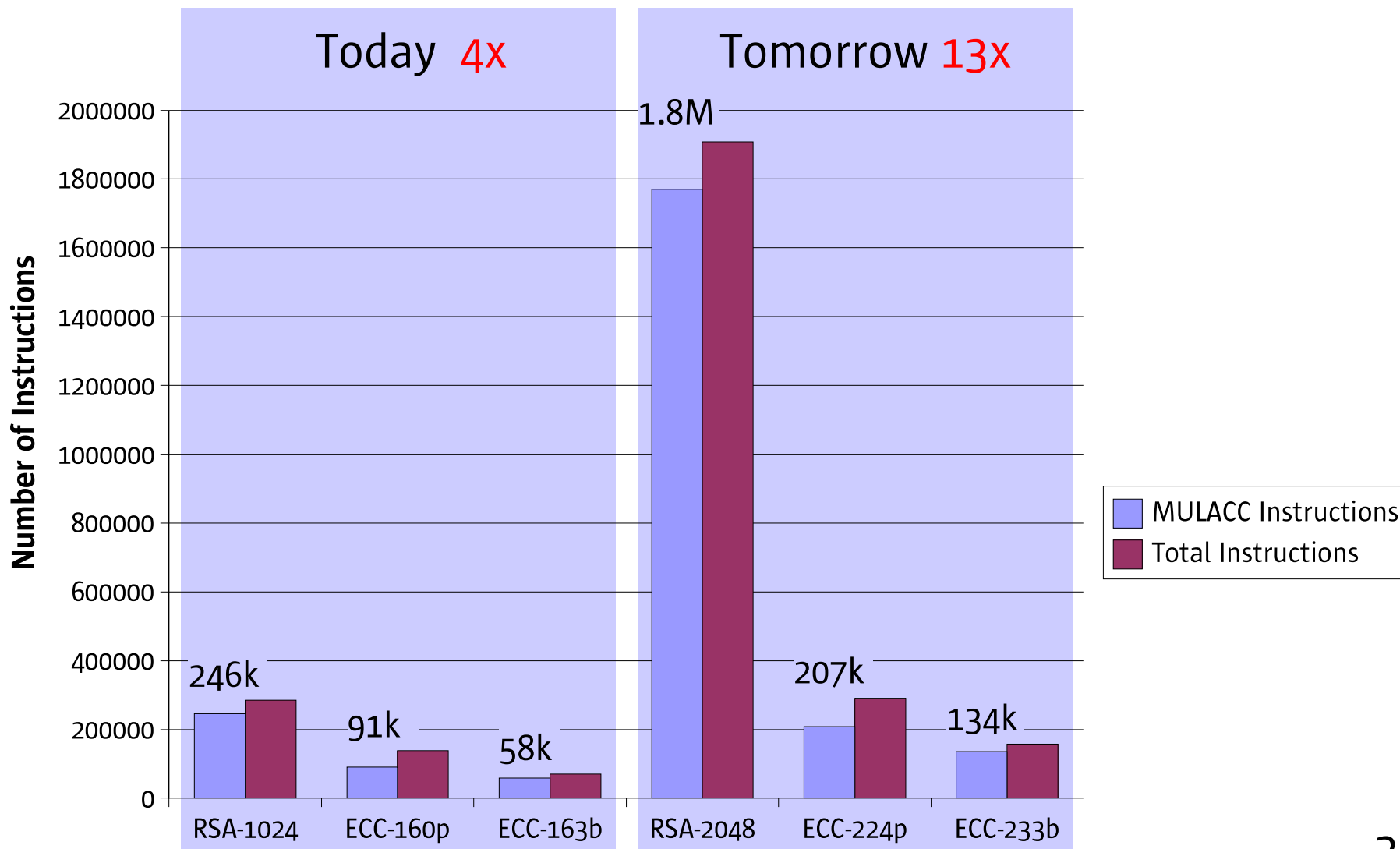
$$u = X * Y * 2^{-k} \pmod{P}$$



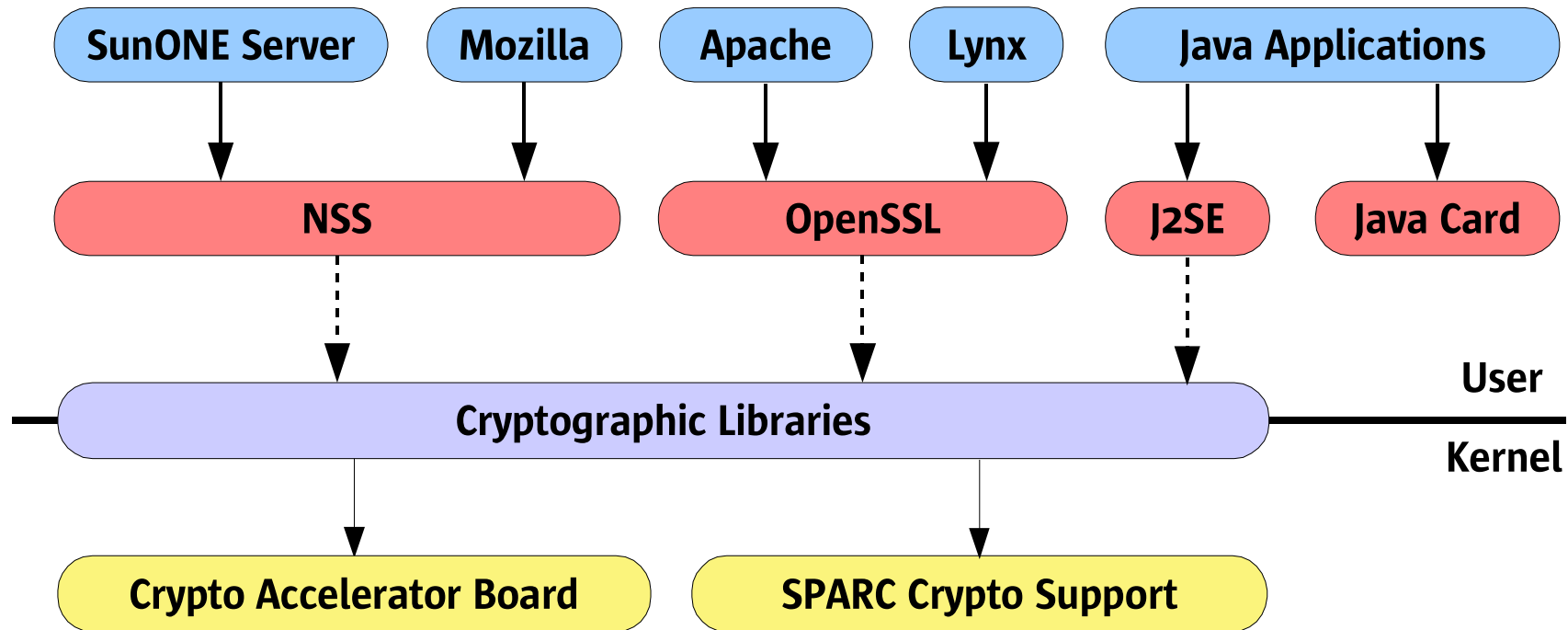
...



Complexity Analysis



Enable the Internet with the Next Generation Crypto Technology



Security is no longer an option, it is becoming an **integral part** of a system architecture.



www.research.sun.com/projects/crypto

Vipul Gupta
vipul.gupta@sun.com

Hans Eberle
hans.eberle@sun.com



ECC Integration Details (NSS)

(Inside [mozilla/security/nss/lib/](https://github.com/mozilla/security/nss/lib/))

